
8.1 Introduction

8.1.1 Stages of a CFD Analysis

A complete CFD analysis consists of:

- *pre-processing*;
- *solving*;
- *post-processing*.

This course has focused on the "solving" process, but this is of little use without pre-processing and post-processing programs. Commercial CFD vendors supplement their flow solvers with grid-generation and flow-visualisation tools. These are specialist areas in their own right, with much money and effort expended on developing "user-friendly" interfaces to make CFD generally accessible and to facilitate its application to very complex flows

<u>Pre-Processing</u>
The pre-processing stage consists of:
–determining the *equations* to be solved;
– specifying the *boundary conditions*;
– generating a computational *mesh*.
It depends upon:
– the desired outcome of the simulation (e.g. forces, loss coefficients, flow rate, concentration distribution, heat transfer, ...);
– the capabilities of the solver.

<u>Solving</u>
In commercial CFD packages the solver is often operated as a "black box". Nevertheless, intelligent user intervention is necessary – to set under-relaxation factors and input parameters, for example – whilst an understanding of discretisation methods and internal data structures is necessary in order to supply mesh data in an appropriate form and to analyse the output.

<u>Post-Processing</u>
The raw output of the solver is a set of numbers corresponding to the values of each field variable (u, v, w, p, …) at each point of the mesh. This huge quantity of numbers must be reduced to some meaningful subset and, usually, manipulated further to obtain the desired predictive quantities. For example, a set of surface pressures and cell-face areas is required to compute a drag coefficient or a set of velocities and areas to determine a flow rate. Commercial packages often provide post-processing facilities to plot, interpolate or simply extract quantities from the output dataset. A key component of post-processing is being able to visualise complex flows – either to indicate important features of the flow or, unfortunately, sometimes to establish why a calculation is diverging.

8.1.2 Commercial CFD

The table below lists some of the more popular commercial CFD packages.

Developer/distributor	Code	Web address
Fluent	FLUENT	http://www.fluent.com/
CD adapco	STAR-CD	http://www.cd-adapco.com/
Ansys	CFX	http://www-waterloo.ansys.com/cfx/
Ansys	ANSYS	http://www.ansys.com/
Flow Science	FLOW3D	http://www.flow3d.com/
CHAM	PHOENIX	http://www.cham.co.uk/

An excellent web portal for all things CFD is http://www.cfd-online.com/.

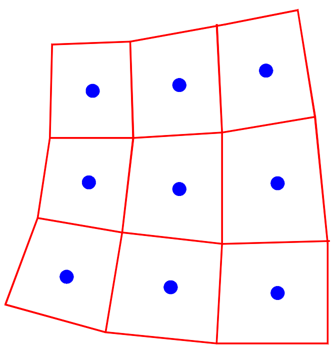
8.2 The Computational Mesh

8.2.1 Mesh Structure

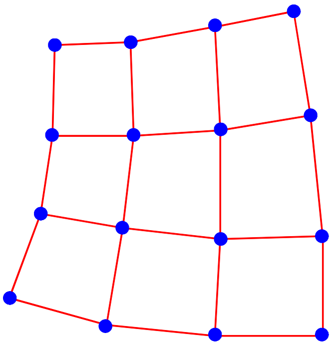
The output of the mesh generator is determined by the discretisation method and the way that the flow solver reads and stores geometric information.

Finite-difference methods, which discretise the differential form of the governing equations, require a structured lattice of points at which flow variables are to be stored.

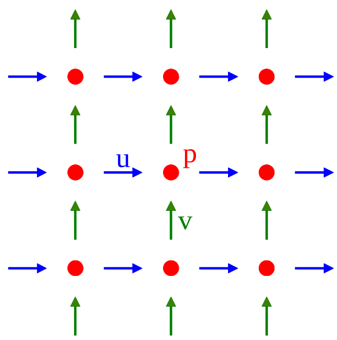
Finite-volume methods require that the *vertices* of control volumes be specified. Precisely where variables are stored relative to these vertices depends on the method employed: for example, cell-centred or cell-vertex. Further complexity is introduced if a staggered velocity grid is employed.



cell-centred storage

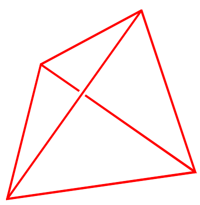


cell-vertex storage

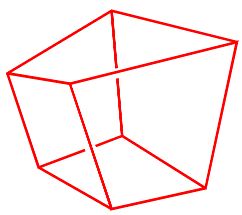


staggered velocity mesh

The *shapes* of control volumes depend on the capabilities of the solver. Structured-grid codes use quadrilaterals in 2-d and hexahedra in 3-d flows. Unstructured-grid solvers often use triangles (2-d) or tetrahedra (3-d).



tetrahedron



hexahedron

In all cases it is necessary to specify *connectivity*; that is, which nodes are adjacent to each other, which nodes are situated either side of a particular face and which faces form the boundaries of a particular finite volume. For structured grids, with (i,j,k) numbering this is straightforward, but for unstructured grids quite complicated data structures must be set up to store connectivity information.

8.2.2 Areas and Volumes (**** *MSc Course Only* ****)

In continuum mechanics, conservation laws take the form:

rate of change + net flux = source

To calculate fluxes requires the (vector) areas of cell faces. To find the total amount of some property in a control volume requires its volume.

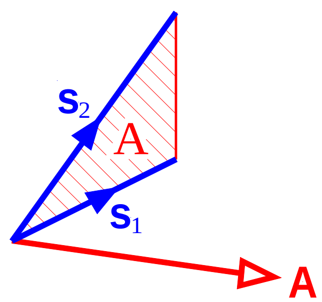
Areas and volumes are easy to evaluate for a cartesian mesh, but general-purpose CFD requires more advanced geometrical techniques.

Areas

Triangles.
The vector area of a triangle is

$$\mathbf{A} = \tfrac{1}{2}\mathbf{s}_1 \times \mathbf{s}_2$$

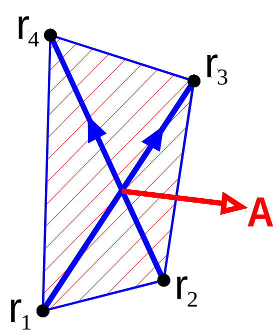
where $\mathbf{s}_1$ and $\mathbf{s}_2$ are side vectors. The orientation depends on the order of vectors in the cross product.



Quadrilaterals
4 points do not, in general, lie in a plane. However (see the example sheet), the vector area of *any* surface spanned by these points and bounded by the side vectors is equal to half the cross product of the diagonals:

$$\mathbf{A} = \tfrac{1}{2}\mathbf{d}_{13} \times \mathbf{d}_{24} = \tfrac{1}{2}(\mathbf{r}_3 - \mathbf{r}_1) \times (\mathbf{r}_4 - \mathbf{r}_2)$$

(Again, the order of points determines the orientation of the area vector).



Volumes

Since $\nabla \bullet \mathbf{r} = 3$, where $\mathbf{r}$ is the position vector, integrating over an arbitrary control volume and using the divergence theorem gives

$$V = \frac{1}{3}\oint_{\partial V} \mathbf{r} \bullet \mathrm{d}\mathbf{A}$$

where d$\mathbf{A}$ is aligned along the *outward* normal.

If the volume has plane faces this can be evaluated as

$$V = \frac{1}{3}\sum_{\substack{plane\\faces}} \mathbf{r}_f \bullet \mathbf{A}_f$$

where $\mathbf{r}_f$ is any convenient position vector on a face and $\mathbf{A}_f$ is its vector area, since, for any other vector $\mathbf{r}$ in that face,

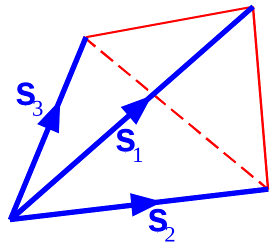
$$\mathbf{r} \bullet \mathrm{d}\mathbf{A} = \mathbf{r}_f \bullet \mathrm{d}\mathbf{A} + \underbrace{(\mathbf{r} - \mathbf{r}_f) \bullet \mathrm{d}\mathbf{A}}_{=0}$$

(The last term vanishes because $\mathbf{r} - \mathbf{r}_f$ is perpendicular to d$\mathbf{A}$ for any point on that plane face.)

Tetrahedra

The volume of a tetrahedron formed from side vectors $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$ (taken in a right-handed sense) is

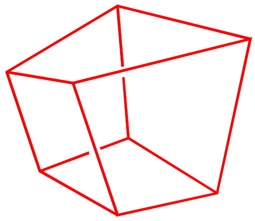
$$V = \tfrac{1}{6}\mathbf{s}_1 \bullet \mathbf{s}_2 \times \mathbf{s}_3$$



Hexahedra

Volumes of arbitrary hexahedral cells are taken as:

$$V = \frac{1}{3}\sum_{faces} \mathbf{r}_c \bullet \mathbf{A}$$



where, for each face, the reference point is taken at the centre:

$$\mathbf{r}_c = \tfrac{1}{4}(\mathbf{r}_1 + \mathbf{r}_2 + \mathbf{r}_3 + \mathbf{r}_4)$$

and the vector area is

$$\mathbf{A} = \tfrac{1}{2}(\mathbf{r}_1 - \mathbf{r}_3) \times (\mathbf{r}_2 - \mathbf{r}_4)$$

Points $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, $\mathbf{r}_4$ should be taken in a clockwise order when viewed along the outward normal.

This is a generalisation to arbitrary hexahedra of the result for cartesian control volumes.

Volume-Averaged Derivatives

By applying the divergence theorem to $\phi\mathbf{e}_x$, where $\mathbf{e}_x$ is the unit vector in the x direction it may be shown that the volume-averaged derivative of a scalar field ϕ is

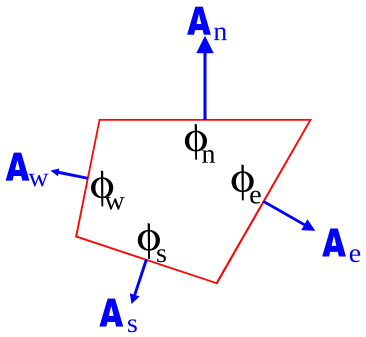
$$\left\langle \frac{\partial \phi}{\partial x} \right\rangle = \frac{1}{V}\int_{\partial V} \frac{\partial \phi}{\partial x}\,\mathrm{d}V = \frac{1}{V}\oint_{\partial V} \phi\, \mathrm{d}A_x$$

Thus for non-cartesian – and, in particular, non-hexahedral – cells, average derivatives may be derived from the values of ϕ on the cell faces, together with the components of the face area vectors and the cell volume.

e.g.

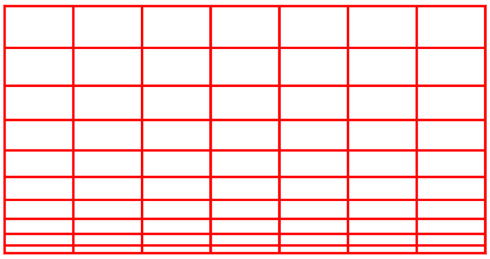
$$\frac{\partial \phi}{\partial x} = \frac{1}{V}(\phi_w A_{wx} + \phi_e A_{ex} + \phi_s A_{sx} + \phi_n A_{nx} + \phi_b A_{bx} + \phi_t A_{tx})$$

where the values on cell faces, ϕ_w, ϕ_e etc., are obtained by interpolation from the nodes on either side.



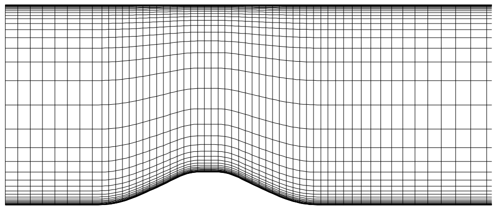
8.2.3 Classification of Grid Types

Grids can be *cartesian* or *curvilinear* (*body-fitting*). In the former, grid lines are always parallel to the coordinate axes. In the latter, coordinate surfaces are curved to fit boundaries. There is an alternative division into *orthogonal* and *non-orthogonal* grids. In orthogonal grids (for example, cartesian or polar meshes) all grid lines cross at 90°. Some flows can be treated as *axisymmetric*, and in these cases, the flow equations can be expressed in terms of *polar coordinates* (r,θ), rather than cartesian coordinates (x,y), with minor modifications.



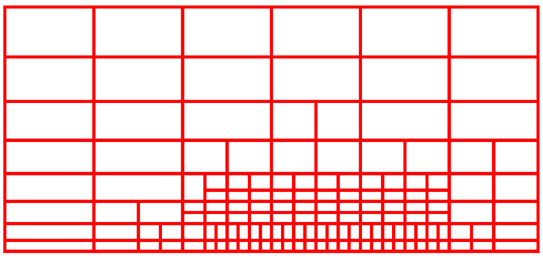
single-block structured cartesian mesh

Structured grids are those whose control volumes can be indexed by (i,j,k) for $i = 1,..., n_i$, $j = 1,.., n_j$, $k = 1,..., n_k$, or by sets of such blocks (*multi-block structured grids* – see below). Each structured block of control volumes, even if curvilinear, can be distorted by a coordinate transformation into a cube (or square in two dimensions). Multi-block structured meshes can accommodate most practical flow configurations.

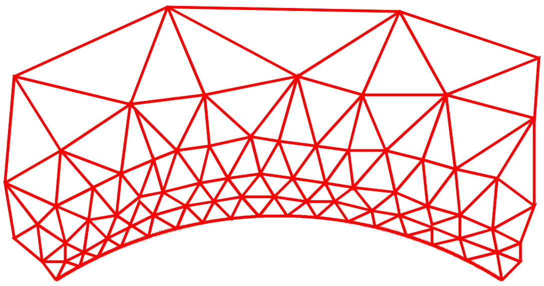


single-block structured curvilinear mesh

Unstructured meshes can accommodate completely arbitrary geometries. However, there are significant penalties to be paid for this flexibility, both in terms of the connectivity data structures and solution algorithms. Grid generators and plotting routines for such meshes are also very complex.



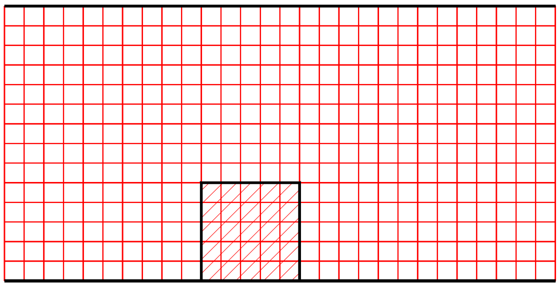
unstructured cartesian mesh



unstructured triangular mesh

8.2.4 Fitting Complex Boundaries

Blocking Out Cells



mesh for 2-d rib with blocked-out cells

The range of flows which can be computed in a rectangular domain is rather limited. Nevertheless, a number of significant bluff-body flows can be computed using a single-block cartesian mesh by the process of *blocking out* cells. This, in fact, was how the solver handled the surface-mounted rib in the demonstration program. Solid-surface boundary conditions are applied to cell faces abutting the blocked-out region, whilst values of velocity and other flow variables are forced to zero by a modification of the source term for those cells. In the notation of Section 4, where the scalar-transport equations for a single cell are discretised as

$$a_P \phi_P - \sum a_F \phi_F = b_P + s_P \phi_P$$

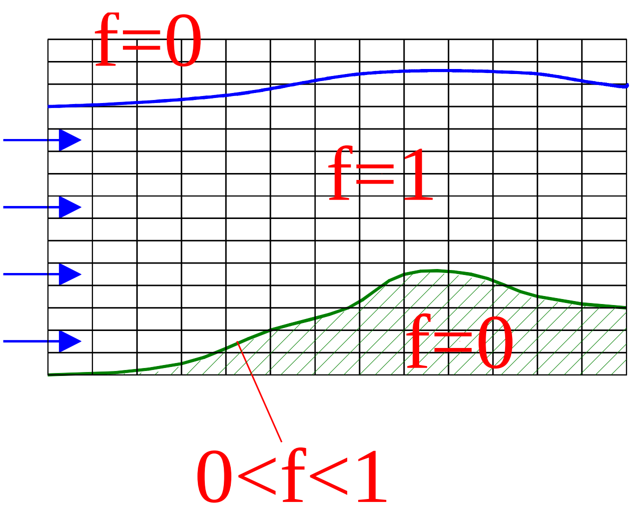
the source terms are simply re-set:

$$b_P \to 0, \quad s_P \to -\gamma$$

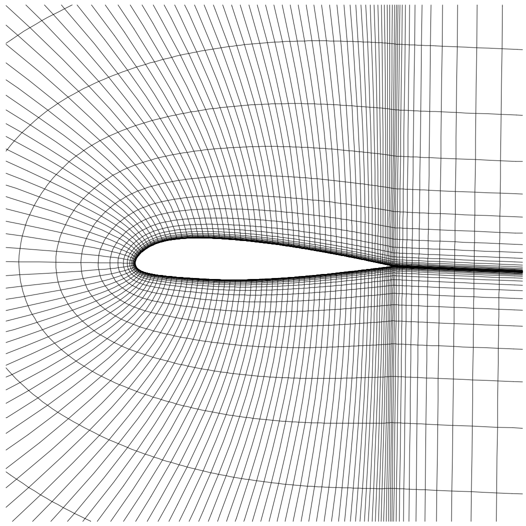
where γ is a large number (e.g. 10^{30}). Rearranging for ϕ_P, this ensures that the computational variable ϕ_P is effectively forced to zero in these cells. However, the computer still stores and carries out operations for these points, so that it is essentially performing a lot of redundant work. An alternative approach is to fit several structured grid blocks around the bluff body. Multi-block grids will be discussed further below.

Volume-of-Fluid Approach

The numerical simplicity and solver efficiency accruing from a cartesian grid mean that some practitioners still attempt to retain this grid geometry even for complex curved boundaries: both for solid walls and free surfaces. In this *volume-of-fluid* approach the fraction f of the cell filled with fluid is stored: 0 outside the fluid domain, 1 within the interior of the fluid and $0 < f < 1$ for cells which are cut by a boundary. At solid boundaries f is determined by the surface contour. At free surfaces, f emerges as part of the solution procedure.



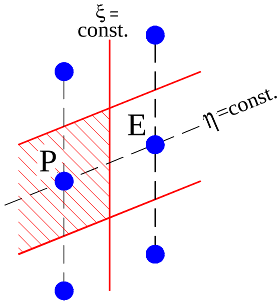
Body-Fitted Grids



The majority of general-purpose codes employ body-fitted (curvilinear) grids. The mesh lines/coordinate surfaces are distorted so as to fit snugly around curved boundaries. Accuracy in turbulent-flow calculations demands a high density of grid cells close to solid surfaces and the use of body-fitted meshes means that the grid need only be refined in the direction normal to the surface, with consequent saving of computer resources.

However, the use of body-fitted grids has important consequences.

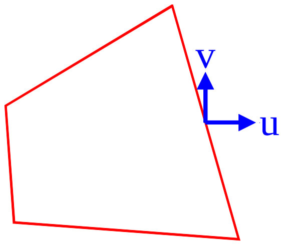
- It is necessary to store detailed geometric components for each control volume; for example, in our research code STREAM we need to store (x,y,z) components of the cell-face-area vector for "east", "north" and "top" faces of each cell, plus the volume of the cell itself – a total of 10 arrays.



- Unless the mesh happens to be orthogonal, the diffusive flux through the east face (say) is no longer given exactly by

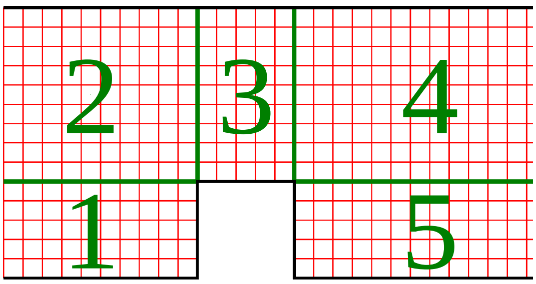
$$\Gamma \frac{\partial \phi}{\partial n} A = \Gamma \frac{\phi_E - \phi_P}{\Delta_{PE}} A$$

  because the discretised derivative of ϕ normal to the face involves *cross-derivative* terms parallel to the cell face and nodes other than P and E. The extra off-diagonal diffusion terms are typically transferred to the source term.



- A similar alignment problem occurs with the advection terms because, in general, interpolated values of all three velocity components are needed to evaluate the mass flux through a single face. This necessitates approximations in the pressure-correction equation that can slow down the solution algorithm.

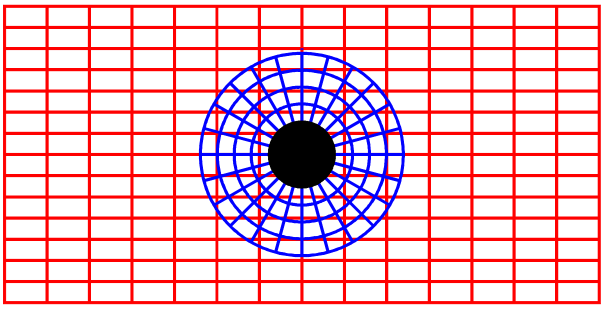
8.2.5 Multi-Block Structured Grids



2-d rib with multi-block mesh

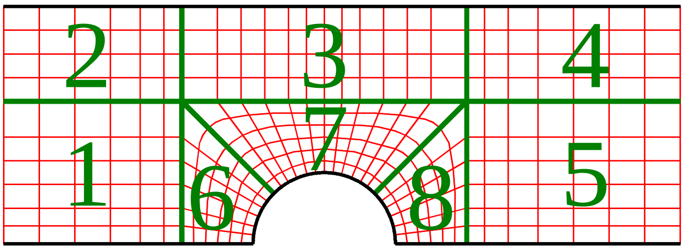
In multi-block structured grids the domain is decomposed into a small number of regions, in each of which the mesh is structured (i.e. cells can be indexed by (i,j,k)).

A common arrangement (and that assumed by our own code STREAM), is that grid lines match at the interface between two blocks, so that there are cell vertices that are common to two blocks. However, some solvers do allow overlapping blocks (*chimera grids*) or block boundaries where cell vertices do not align. Interpolation is then needed at the boundaries of blocks.



chimera grid

In the usual arrangement, with both cell vertices matching at block boundaries and blocks meeting "whole-face to whole-face" the solver starts by adding additional lines of cells from one block to those of the adjacent, so that, in discretising, accuracy is not compromised. On each iteration of a scalar transport equation the discretised equations would be solved implicitly within each block, with values from the adjacent block providing boundary conditions. At the end of each iteration, values in overlap cells would be explicitly updated using data from the interior of the adjacent block.



Multiple blocks are employed to maintain a structured grid configuration around complex boundaries. There are no hard and fast rules, but it is generally desirable to avoid sharp changes in grid direction (which lead to lower accuracy) in important and rapidly changing regions of the flow, such as near solid boundaries. One should also strive to minimise the non-orthogonality of the grid.

8.2.6 Disposition of Grid Cells

Only two points are necessary to resolve a straight-line profile, but the more curved the profile the more points are needed to resolve it accurately. In general, more points are needed in rapidly-changing regions of the flow, such as:

- solid boundaries;
- separation, reattachment and impingement points;
- flow discontinuities; e.g. shocks, hydraulic jumps.

Simulations must demonstrate *grid-independence*, i.e. that a finer-resolution grid would not significantly modify the solution? This generally requires a sequence of calculations on successively finer grids.

Boundary conditions for turbulence modelling do impose some limitations on the cell size near walls. Low-Reynolds-number models (integrating through the semi-viscous sublayer to the wall) generally require that $y^+ < 1$ for the near-wall node, whilst high-Reynolds-number models relying on wall functions strictly require the near-wall node to lie in the log-law region, say $15 < y^+ < 150$. (Here, $y^+ = yu_\tau/\nu$ is a non-dimensional length formed from the wall distance y and friction velocity $u_\tau = \sqrt{\tau_w/\rho}$).

8.2.7 Multiple Grids

Multiple grids – combining cells so that there are 1, 1/2, 1/4, 1/8, ... times the number of cells in some basic fine grid – are used deliberately in so-called *multigrid* methods. These calculate the solution on alternately coarser and finer grids. The idea is that the solution is obtained quickly on the coarsest grid where the number of cells is small and changes propagate rapidly across the domain. The solution is then refined locally on the finest grid to obtain the most accurate solution.

If two levels of grid are used then a process known as *Richardson extrapolation* may be used to both estimate the error and, possibly, refine the solution. If the basic advection-diffusion discretisation is known to be of order n and the exact (but unknown) solution of some property ϕ is denoted ϕ^*, then one would expect the error $\phi - \phi^*$ to be proportional to Δ^n, where Δ is the mesh spacing. Thus, for solutions ϕ_Δ and $\phi_{2\Delta}$ respectively on two grids with mesh spacing Δ and 2Δ,

$$\begin{aligned} \phi_\Delta &= \phi^* + C\Delta^n \\ \phi_{2\Delta} &= \phi^* + C(2\Delta)^n \end{aligned}$$

Two equations give us two equations for two unknowns, ϕ^* and C, which we can solve to get a better estimate of the solution:

$$\phi^* = \frac{2^n \phi_\Delta - \phi_{2\Delta}}{2^n - 1}$$

and the error in the fine-grid solution:

$$C\Delta^n = \frac{\phi_{2\Delta} - \phi_\Delta}{2^n - 1}$$

Unstructured-grid methods offer the possibility of *local grid refinement*: that is, adding more cells in regions where the error is estimated (e.g. by Richardson extrapolation) to be high. For structured-grid topologies, however, this would usually require generation of a completely new mesh.

8.3 Boundary Conditions

The number and type of boundary conditions must accord with the governing equations of the flow, which, for the elliptic equations describing steady incompressible flow, means a condition on each transported flow variable around the whole boundary of the flow domain.

There are a number of common boundary conditions on transported variables ϕ:

- *inflow* boundary conditions: value specified on the boundary, either by a predefined profile or doing an initial 1-d fully-developed-flow calculation;
- *outflow* boundary conditions: usually $\partial\phi/\partial n = 0$, where n is the direction normal to the outflow boundary;
- *wall* boundary conditions: zero velocity plus wall stress by viscous-stress or wall-function expressions;
- *symmetry plane*: $\partial\phi/\partial n = 0$, except for the velocity component normal to the boundary, which would be set to zero;
- *periodic* boundary conditions.

8.4 Flow Visualisation

CFD has a reputation for producing colourful output and, whilst some of it is promotional, the ability to display results effectively may be an invaluable design tool.

8.4.1 Available Packages

Visualisation tools are often packaged with commercial CFD products. However, many excellent stand-alone applications or libraries are available. Some of the more popular in CFD are listed below. Most have versions for various platforms, although some run only on certain versions of unix. Some are aimed predominantly at the CFD user, but others are general-purpose visualisation tools, which may equally well be applied in other branches of engineering. Some are even free!

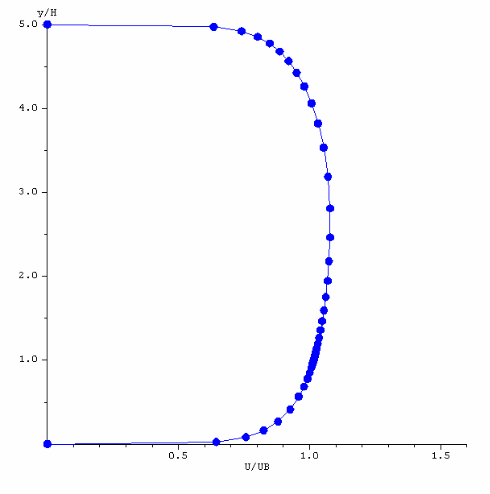
- TECPLOT (Amtec Engineering) http://www.amtec.com/
- AVS, Gsharp, Toolmaster (Advanced Visual Systems) http://www.avs.com/
- PV-WAVE (Visual Numerics) http://www.vni.com/index.html
- Iris Explorer (NAG) http://www.nag.co.uk/visualisation_graphics.asp
- OpenDx (IBM) http://www.opendx.org/ (free, apparently)

But for simple line, contour and vector plots, many researchers are still happy with the original free, open-source plotting package ...

- Gnuplot (open-source software community) http://www.gnuplot.info/

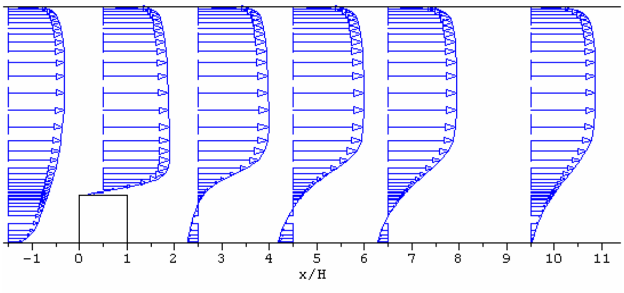
8.4.2 Types of Plot

x-y plots

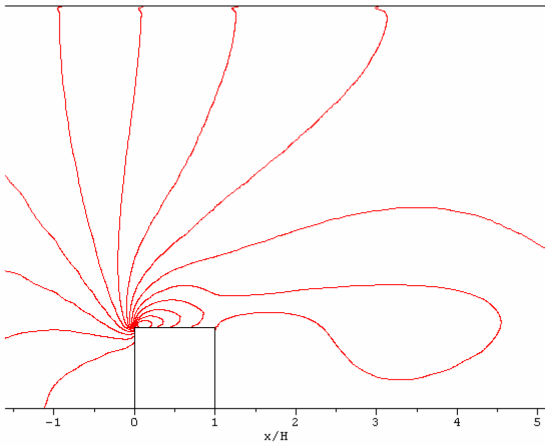


These are simple, two-dimensional graphs. They can be drawn by hand or by many plotting packages. They are the most precise and quantitative way to present numerical data and, since laboratory data is often gathered by straight-line traverses, they are a popular way of making a direct comparison between experimental and numerical data. Logarithmic scales also allow the identification of important effects occurring at very small scales, particularly near solid boundaries. They are widely used for line profiles of velocity and stresses and for plots of surface quantities such as pressure and skin-friction coefficient.

One way of visualising the development of the flow is to use several successive profiles.



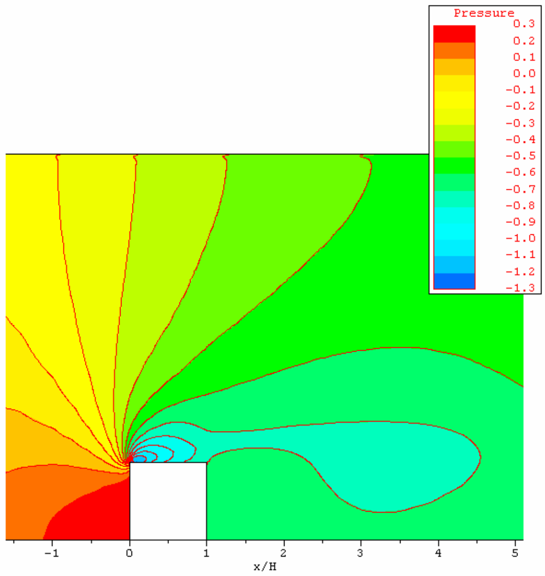
Line Contour Plots



A contour line ("isoline") is a line along which some property is constant. The equivalent in 3 dimensions is an "isosurface". Any field variable may be contoured. In contrast to line graphs, contour plots give a global view of the flow field, but are less useful for reading off precise numerical values. If the domain is linearly scaled then detail occurring in small regions is often obscured.

The actual numerical values of the isolines are sometimes less important than their overall disposition. If contour intervals are the same then clustering of lines indicates rapid changes in flow quantities. This is particularly useful in locating shocks and discontinuities.

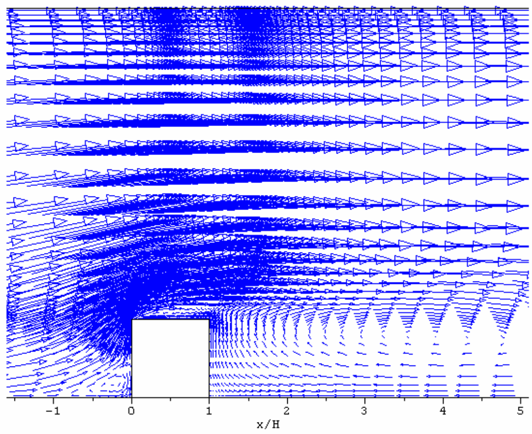
Shaded Contour Plots



WARNING: colour plots are difficult and expensive to print or photocopy! This proviso apart, colour is an excellent medium for conveying information and good for on-screen and presentational analysis of data. Simple packages flood the region between isolines with a fixed colour for that interval. The most advanced packages allow a pixel-by-pixel gradation of colour between values specified at the cell vertices, together with lighting and other special effects such as translucency.

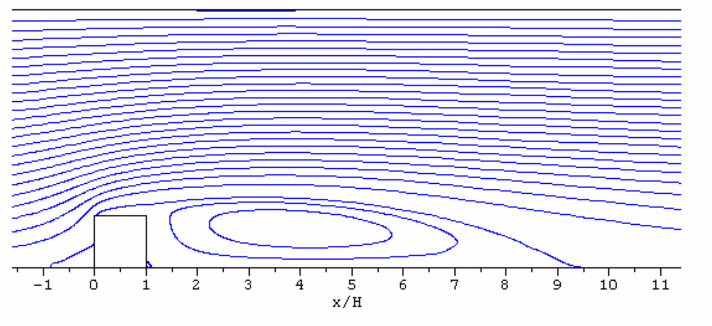
Grey-scale shading is an option if plots are to be reproduced in black and white.

Vector Plots



Vector plots display vector quantities (usually velocity; occasionally stress) with arrows whose orientation indicates irection and whose size (and sometimes colour) indicates magnitude. They are a popular and informative means of illustrating the flow field in two dimensions, although if grid densities are high then either interpolation or reduced numbers of output positions are necessary to prevent the number density of arrows blackening the plot. There can sometimes be problems when selecting scales for the arrows when large velocity differences are present, especially in important areas of recirculation where the mean flow speed is low. In three dimensions, vector plots can be deceptive because of the angle from which they are viewed.

Streamline Plots



Streamlines are parallel to the mean velocity vector. They can generally be obtained by integration:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}$$

(the only option in three dimensions), but a more accurate method in 2-d is to contour the streamfunction ψ. This can be derived from the mean velocity field in 2-d by

$$\frac{\partial \psi}{\partial y} = u\,, \quad \frac{\partial \psi}{\partial x} = -v$$

but, more effectively by using the property that

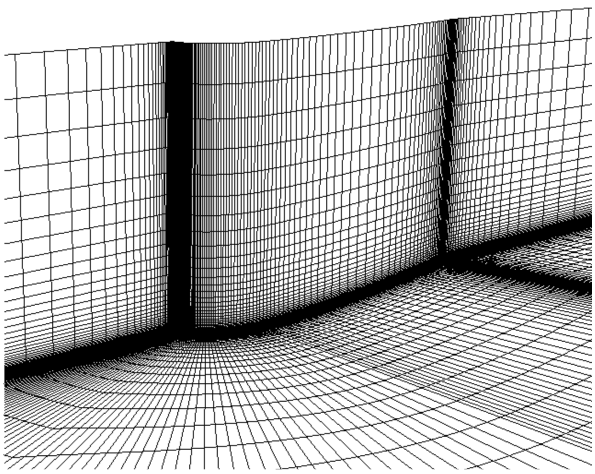
$$\int_1^2 \mathbf{u} \bullet \mathbf{n}\, \mathrm{d}s = \psi(2) - \psi(1)$$

which is simply the volume flow rate across any line connecting points 1 and 2. Thus, ψ is well defined for any 2-d incompressible velocity field satisfying continuity.

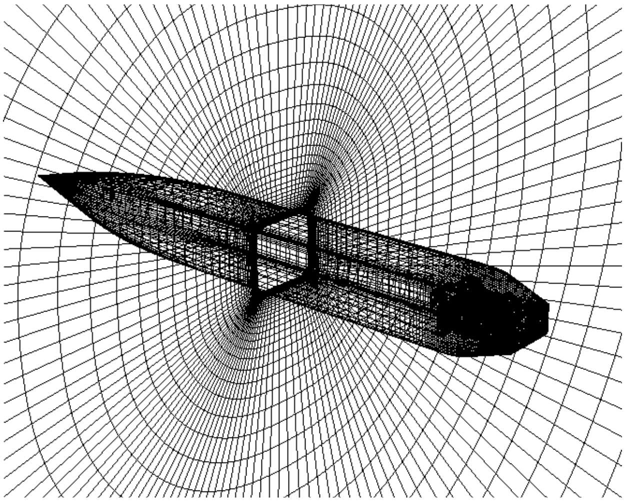
If isolines are equally spaced in values of ψ, then clustering of lines corresponds to high velocities and regions where they are further apart signify low velocities. However, as with vector plots, this has the effect of making it difficult to visualise the actual flow pattern in low-velocity regions such as recirculation zones.

Particle paths can also be traced along solid surfaces using the wall stress vector. This often reveals important features connected with separation/reattachment/impingement on 3-d surfaces, which defines the basic flow topology.

Mesh Plots

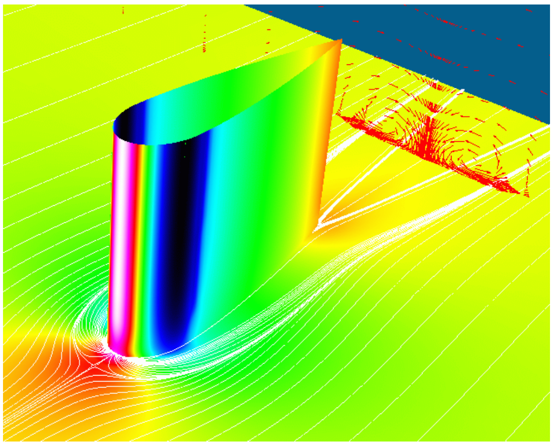


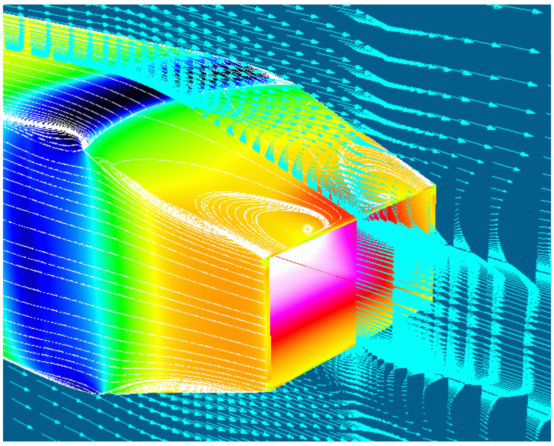
The computational mesh is usually visualised by plotting lines corresponding to the edges of control volumes – for one coordinate surface only in 3 dimensions. It can be very difficult indeed to visualise fully-unstructured 3-d meshes, and usually only the surface mesh is portrayed.



Composite Plots

In 3 dimensions it is common to combine plots of the above types, emphasising the parallel variation of important quantities. In this way, some sense can be made of the highly complex flows now being studied with CFD.





Finally, just to put things in perspective ...

> "*On two occasions I have been asked* [by members of Parliament], '*Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out*'. *I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.*"
>
> Charles Babbage

(Robert Peel's government eventually pulled the plug on Babbage's research and he was unable to build his "analytical engine". Ever since then the British Government has preferred to fight wars on the other side of the world rather than funding scientific research!)